(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 009 189 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
14.06.2000 Bulletin 2000/24

(51) Int. Cl.^{7}: H04Q 11/04, H04L 12/56

(21) Application number: 99112183.1

(22) Date of filing: 24.06.1999

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: 08.12.1998 US 206975

(71) Applicant: NEC CORPORATION
Tokyo (JP)

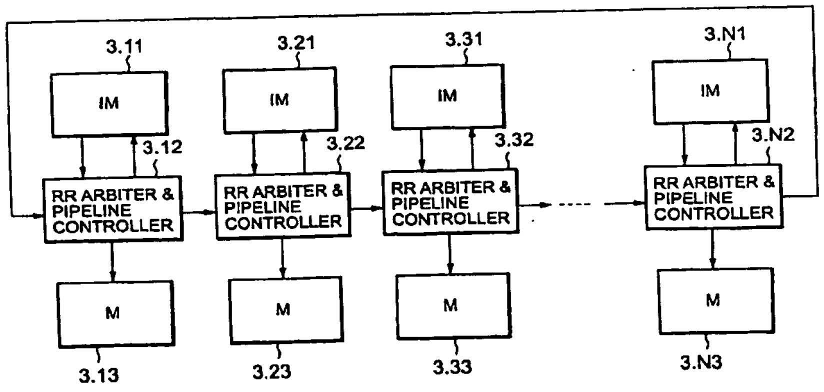
(72) Inventors:
- Ramamurthy, Gopalakrishnan,
c/o NEC USA, Inc.,
Princeton, New Jersey 08540 (US)
- Fan, Ruixue,
c/o NEC USA, Inc.,
Princeton, New Jersey 08540 (US)
- Smiljanic, Aleksandra,
c/o NEC USA, Inc.,
Princeton, New Jersey 08540 (US)

(74) Representative:
Baronetzky, Klaus, Dipl.-Ing. et al
Splanemann Reitzner
Baronetzky Westendorp
Patentanwälte
Rumfordstrasse 7
80469 München (DE)

(54) RRGS-round-robin greedy scheduling for input/output buffered terabit switches

(57) A novel protocol for scheduling of packets in high-speed cell based switches is provided. The switch is assumed to use a logical cross-bar fabric with input buffers. The scheduler may be used in optical as well as electronic switches with terabit capacity. The proposed round-robin greedy scheduling (RRGS) achieves optimal scheduling at terabit throughput, using a pipeline technique. The pipeline approach avoids the need for internal speedup of the switching fabric to achieve high utilization. a method for determining a time slot in a NxN crossbar switch for a round robin greedy scheduling protocol, comprising N logical queues corresponding to N output ports, the input for the protocol being a state of all the input-output queues, output of the protocol being a schedule, the method comprising: choosing input corresponding to i = (constant-k-1) mod N, stopping if there are no more inputs, otherwise choosing the next input in a round robin fashion determined by i = (i + 1) mod N; choosing an output j such that a pair (i,j) to a set C= {(i,j) | there is at least one packet from I to j} , if the pair (i,j) exists; removing i from a set of inputs and repeating the steps if the pair (i,j) does not exist; removing i from the set of inputs and j from a set of outputs; and adding the pair (i,j) to the schedule and repeating the steps.

FIG.3



EP 1 009 189 A2

Description

[0001] This invention relates to terrabit switches for use in applications like electronic and optical media. Specifically, this invention relates to a round robin greedy scheduling algorithm. This invention is embodied in methods for scheduling and a terrabit switching system that implements the round robin greedy scheduling algorithm.

[0002] With the growing demand for bandwidth there is an increasing need for terabit switching. *see* M. Beshai, and E. Miinter, "Multi-tera-bit/s switch based on burst transfer and independent shared buffers," *ICC'95*, pp. 1724-1730, N. McKeown et *al.*, "The tiny tera: a packet switch core," *IEEE Micro*, vol. 171, Jan.-Feb. 1997, pp. 26-33, and W. D. Zhong, Y. Shimazu, M. Tsukuda, and K. Yukimatsu, "A modular Tbit/s TDM-WDM photonic ATM switch using optical buffers," *IEICE Transactions on Communications*, vol. E77-B, no. 2, February 1994, pp. 190-196. Optical switching core (which can be viewed as a logical cross bar switch) with electronic control is an attractive candidate for high capacity switches. At a line rate of 10 Gb/s, a 64 byte cell/packet has to be processed within 40 ns.

[0003] An important issue faced by practitioners in the field is how to make fast scheduling decisions that will use the optical core efficiently. Switch design in such a case may involve input buffering, output buffering or both. In a switch with output buffering, the output buffers require access speed greater than the total switch throughput. Alternatively, a knockout architecture is employed in order to decrease the required output buffer speed, where a limited number of cells are accepted by the output buffer and the rest are dropped. An optical knockout switch has been proposed in, Zhong, Y. Shimazu, M. Tsukuda, and K. Yukimatsu, "A modular Tbit/s TDM-WDM photonic ATM switch using optical buffers," *IEICE Transactions on Communications*, vol. E77-B, no. 2, February 1994, pp. 190-196. The complexity of optical knockout switching is high since each output requires several optical reverse Banyan networks and optical buffers.

[0004] Switches with input buffering use buffers more efficiently, and need memory bandwidth of only twice the line rate. In a simple scheme with input buffering, all inputs make requests to transmit packets that are at the head of their queues. If two or more inputs make requests for the same output, one of them is chosen randomly. It was shown in, see M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input vs. output queuing on a space- division packet switch," *IEEE Transactions on Communications*, vol. COM-35, no. 12, December 1987, pp. 1347-1356 that input buffering algorithm leads to a throughput of 0.587 under uniform traffic condition. The efficiency further decreases in the case of non-uniform traffic. In several other scheduling schemes, packets other than the HOL (head of line) packets contend for output ports. See R. Fan, M. Akiyama, and Y. Tanaka, "An input buffer-type ATM switching using schedule comparison," *Electronics and Communications in Japan: Part I*, vol. 74, no. 11, 1991, pp.17-25; S. Motoyama, D. W. Petr, and V. S. Frost, "Input-queued switch based on a scheduling algorithm," *Electronics Letters*, vol. 31, no. 14, July 1995, pp. 1127-1128; and H. Obara, "Optimum architecture for input queuing ATM switches," *Electronics Letters*, vol. 27, no. 7, March 1991, pp. 555-557. During each time slot, an input issues requests to several outputs. With just 4 requests per time slot, an efficiency approaching 1 is achieved. However, in such a scheme at high speeds multiple request/acknowledgements for scheduling cannot be processed within one time slot (where a slot represents a packet transmission time). Also, for the non-uniform traffic with hot-spots, the performance may degrade since inputs independently decide which outputs they are going to request.

[0005] The switch performance can be improved if the switch controller knows the states of all input-output queues. Such information enables the switch controller to increase the number of simultaneous transmissions in each time slot. In the SLIP protocol, the outputs independently issue grants to the inputs, which leads to some inefficiency, *see* N. McKeown, P. Varaiya, J. Walrand, "Scheduling cells in an input-queued switch," *Electronic Letters*, vol. 29, no. 25, December 1993, pp. 2174-2175. Better coordination between inputs is achieved by the algorithms discussed in, *see* D. Guo, Y. Yemini, Z. Zhang, "Scalable high-speed protocols for WDM optical star networks," *IEEE INFOCOM'94*. However, these algorithms have a disadvantage that they require many time slots to make scheduling decisions.

[0006] It is an objective of the present invention to solve the above-mentioned problems in the conventional technologies. Specifically it is an objective of the present invention to provide a method for making scheduling decisions in a terabit switch that will efficiently use the optical core. It is a further objective of the present invention to provide a pipeline architecture that performs a round-robin greedy scheduling while providing good performance and fulfilling stringent timing requirements without internal speedups.

[0007] In order to meet the above objectives there is provided a method for determining a time slot in a NxN crossbar switch for a round robin greedy scheduling protocol, comprising N logical queues corresponding to N output ports, the input for the protocol being a state of all the input-output queues, output of the protocol being a schedule, the method comprising: choosing input corresponding to i = (constant-k-1) mod N, stopping if there are no more inputs, otherwise choosing the next input in a round robin fashion determined by i = (i + 1) mod N; choosing an output j such that a pair (i,j) to a set C= {(i,j) | there is at least one packet from I to j} ,if the pair (i,j) exists removing i from the set of inputs and j from a set of outputs; and adding the pair (i,j) to the schedule and repeating the steps; removing i from a set of inputs and repeating the steps if the pair (i,j) does not exist; Another aspect of this invention is a method of scheduling wherein in each time slot N distinct schedules are in progress simultaneously for N distant future time slots, the method comprising; making a specific future time slot available to input for scheduling in a round-robin fashion; selecting an output for a kth time slot in future, by an input i; starting a schedule for the kth time slot in future; determining a next input (l+1) mod N and sending to the next input remaining outputs that are free to receive packets during the kth time slot.

[0008] Preferably if an input i is the last input to complete a schedule for the kth time slot, it selects an output (if feasible) and sends a modified output set to a next input; and if the input I completes a schedule for the kth time slot it does not send the output set to the next input.

[0009] Still preferably an input that does not receive a modified set of outputs from a previous input starts a new schedule.

[0010] Another aspect of the present invention is a method of pipelined round robin greedy scheduling for odd number of inputs wherein an i th time slot is completed using a process comprising: Initializing k(0,1)=k(1,1)=...k(N-1,1)=0, const = N+1, wherein, k(i,l) > 0 is the time slot for which input i reserves an output in ith time slot, i_l =(const - N - 1) mod N denotes an input that starts a new schedule in *l* th time slot, and k(i,l) = 0 implies that the action of input i in time slot *l* is suppressed; setting O_{l+N}={O,1, . . .,N—1}, k(i_l,*l*) = l+N, *k*(*i*,*l*) = *k*((*i*—1) mod *N*,*l*—1) for 0 ≤ *i* ≤ N—1 and *i* ≠ i_l; choosing one output *j* at input *i*, 0 ≦ is ≦ N—1, 1, in a round-robin fashion from the set O_k*(i,1)* for which it has a packet to send, provided that *k(i,l)* ≠ 0 and excluding j from $O_{k(i,l)}$·; storing the address, at input *i*, 0 ≤ *i* ≤ *N* — 1, of the chosen output in its connection memory at location *k(i, l)* mod *N* and moving a head of line (HOL) packet from a corresponding receive input-output queue to separate transmit input-output queue; forwarding the set $O_{k(i,l)}$ at Input *i*, 0 ≤ *i* ≤ *N* — 1 and *i* ≠ (i_l—2) mod *N*, to the next input *(i* + 1) mod *N;* establishing a cross bar connection between the input *i* 0 ≤ *i* ≤ N — 1 and output whose address is read from location *(l* mod *(N* + 1)) of the input *i*'s connection memory; and transmitting the reserved packet at the head of the scheduled transmit input-output queue i through the switch core for each input i, 0 ≤ *i* ≤ *N* — 1.

[0011] Preferably multicast scheduling is incorporated in round-robin greedy scheduling algorithm wherein multicast packets are stored in a first come first served fashion and have priority over unicast queues and steps in *l* th time slot further comprise: choosing all outputs *j* such that *j* ∈ *O k(i,1)* ∩ BM_i at input *i*, 0 ≤ *i* ≤ *N* — 1 and transmitting HOL multicast packets the chosen outputs in the *k*th time slot, serving unicast queues if $O_{k(i,l)}$ ∩ BM_i is an empty set, otherwise excluding the chosen outputs are from $O_{k(i,l)}$ and BM_i; and deleting HOL multicast packets from the multicast queue If BM_i is empty.

[0012] Another aspect of the present invention is a method of pipelined round robin greedy scheduling for even number of inputs wherein an *l* th time slot is completed using a process comprising: initializing k(0,1)=k(1,1)=...k(N-1,1)=0, const = N+1, wherein, k(i, *l*) > 0 is the time slot for which input i reserves an output in *l* th time slot, i_l=(const - N - 1) mod N denotes an input that starts a new schedule in ıth time slot, and k(i,l) = 0 implies that the action of input i in time slot *l* is suppressed; setting $O_{\imath+N}$={O,1,. . .,N—1}, $k_{(il,l)}$ = *l* +N+1, k(mod(i_l+*l*) mod N, *l*) = k(i_l, *l* -2), and k(i, *l*) = k((i—1) mod *N*, *l* - 1) for 0 ≤ *i* ≤ *N* — 1 and *i* ∉ {i_l, (i_l + 1) mod *N*} and l is not equal to $l_\imath$; choosing one output *j* at input i, 0 ≦ *i* ≦ *N* — 1, in a round-robin fashion from the set O_k*(i, l)* for which it has a packet to send, provided that *k(i, l)* ≠ 0 and excluding j from $O_{k(i,l)}$; storing the address, at input i, 0 ≤ *i* ≤ *N* — 1, of the chosen output in its connection memory at location *k(i, l)* mod *(N* + 1) and moving an HOL packet from a corresponding receive input-output queue to separate transmit input-output queue; forwarding the set $O_{k(i,\ l)}$ at Input *i*, 0 ≤ *i* ≤ *N* — 1 and *i* = (i_l—2) mod *N*,to the next input *(i* + 1) mod *N, wherein* Input (i_l — 2) mod *N* delays the set O_k((i_l—2)modN,1) for one time slot before forwarding it; establishing a cross bar connection between the input *i* 0 ≤ *i* ≤ *N* — 1 and output whose address is read from location (*l* mod (*N* + 1)) of the input *i*'s connection memory; transmitting the reserved packet at the head of the scheduled transmit input-output queue is transmitted through the switch core for each input *i*, 0 ≤ *i* ≤ *N*—1.

[0013] Preferably multicast scheduling is incorporated in round-robin greedy scheduling algorithm wherein multicast packets are stored in a first come first served fashion and have priority over unicast queues and steps in ıth time slot further comprise: choosing all outputs *j* such that *j* ∈$O_{k(i,\imath)}$ ∩ BM_i at input *i*, 0 ≤ *i* ≤ *N* — 1 and transmitting HOL multicast packets to the chosen outputs in the *k*th time slot; serving unicast queues if $O_{k(i,\imath)}$ ∩ BM_i is an empty set, otherwise excluding the chosen outputs are from $O_{k(i,\imath)}$ and BM_i; deleting HOL multicast packets from the multicast queue If BM_i is empty.

[0014] Another aspect of the present invention is a N stage pipeline system for scheduling a NxN switch where a stage i is associated with an input i, said stage i schedules transmission to an output in a future time slot, said future time slot rippling through all stages, wherein all pipeline stages corresponding to inputs are performing scheduling concurrently such that no two inputs choose a same future time slot at a same time, output slots being selected based on a round-robin fashion, wherein when an output is selected by a stage, the output is removed from a free pool of outputs such that a pipeline stage does not pick the output that has already been selected during a time slot.

[0015] The above objectives and advantages of the present invention will become more apparent by describing in detail preferred embodiments thereof with reference to the attached drawings in which:

FIG. 1 shows a timing diagram for a preferred embodiment of a switch controller with N =5;
FIG.2 shows a timing diagram for a preferred embodiment of a switch controller with N =4;
FIG.3 shows an implementation of a control for a NxN switch;
FIG.4 shows a comparison of average packet delays for an implementation of RRGS, RGS, HOL, SLIP and I-TDMA, based on both analytic and simulation results;
FIG. 5 shows a complimentary distribution function of packet delay in I-TDMA, SLIP, RGS and RRGS for fixed offered traffic load of (a)0.8 and (b) 0.9 based on simulation results; and
FIG.6 shows average packet delay under non-uniform traffic load for RRGS, RGS, SLIP and I-TDMA for four groups of queues (a) G1 (b) G2, (c) G3 and (d) overall.

[0016] The pipeline architecture according to the present invention performs a "round-robin greedy scheduling" (RRGS). This is a modification of and an improvement over a random greedy scheduling (RGS), *see* R. Chipalkatti, Z. Zhang, and A. S. Acampora, "Protocols for optical star-coupler network using WDM: performance and complexity study," *IEEE Journal on Selected Areas in Communications,* vol. 11, no. 4, May 1993, pp. 579-589; and D. Guo, Y. Yemini, Z. Zhang, "Scalable high-speed protocols for WDM optical star networks," *IEEE INFOCOM'94.* The protocol of the present invention fulfills stringent timing requirements without any internal speedup, and at the same time preserves the good performance of RGS. *see* D. Guo, Y. Yemini, Z. Zhang, "Scalable high-speed protocols for WDM optical star networks," *IEEE INFOCOM'94.* It achieves close to 100% utilization, and handles non-uniform traffic equally well.

1. Round-Robin Greedy Scheduling (RRGS)

[0017] A preferred embodiment is now described in detail. The protocol used in this invention is called a RRGS protocol. Consider an *NXN* cross-bar switch, where each input port i, $i \in \{0, 1, \ldots, N-1\}$, has N logical queues, corresponding to each of the N outputs. All packets that are received by the switch are fixed size cells. An input to the RRGS protocol is the state of all input-output queues. Such an input can be described by a set C as follows:

$C = \{(i,j)$ / there is at least one packet at input i for output $j\}$.

[0018] The output of the protocol is a schedule associating the inputs to the outputs. Such a set S can be described as follows:

$S = \{(i,j)$ / packet will be sent from input i to output $j\}$.

[0019] It will be clear to a skilled artisan that in each time slot, an input can transmit only one packet, and an output can receive only one packet. Under this condition, a schedule for an arbitrary kth time slot is determined as follows:

Step 1) $I_k = \{0,1,\ldots,N-1\}$ is a set of all inputs, $O_k=\{0,1,\ldots,N-1\}$ is a set of all outputs. Select $i = (const - k - 1) \bmod N$. Such a choice of an input that starts a schedule will enable a simple implementation.

Step 2) If I_k is empty, stop. Otherwise choose the next input i in a round-robin fashion according to $i = (i + 1) \bmod N$.

Step 3) Choose in a round-robin fashion .the output j from O_k such that (i,j) 0 C_k. If such an output does not exist then remove i from I_k and go to step 2.

Step 4) Remove input i from I_k, and output j from O_k. Add (i,j) to S_k. Go to Step 2.

[0020] The above protocol is clearly an improvement over the conventional RGS. *See* D. Guo, Y. Yemini, Z. Zhang, "Scalable high-speed protocols for WDM optical star networks," *IEEE INFOCOM'94*. Also see the DAS algorithm described in R. Chipalkatti, Z. Zhang, and A. S. Acampora, "Protocols for optical star-coupler network using WDM: performance and complexity study," *IEEE Journal on Selccted Areas in Communications*, vol. 11, no. 4, May 1993, pp. 579-589. In the conventional RGS, both input and the corresponding matching output are chosen randomly. However, implementation of such a random selection scheme is difficult in practice. Note that in each time slot, N packets can be transferred from the N inputs to the N outputs.

[0021] The process of scheduling a given time slot in RRGS thus consists of N phases. In each phase of a given time slot (in the future), one input chooses one of the remaining outputs for transmission during that time slot. A phase consists of a request from the input module (IM) to the round-robin (RR) arbiter, RR selection, and acknowledgement from the RR arbiter to the IM. The round-robin order in which inputs choose outputs shifts cyclically at each time slot so that it ensures equal access for all inputs.

2. Pipelined RRGS for Odd Number of Inputs

[0022] At high link speeds (e.g. l0Gb/s), the N phases cannot be completed within one time slot (40ns assuming packet size of 64 bytes). With increasing link speeds, using conventional technologies no more than one phase can be completed in one time slot. To overcome this problem, the present invention uses a pipeline approach, where in each time slot, *N* distinct schedules are in progress simultaneously, for *N* distinct time slots in the future. Each phase of a particular schedule involves just one input. In any given time slot, other inputs simultaneously perform the phases of schedules for other distinct future time slots.

[0023] **Definition:** A schedule for a future time slot T_k is said to be completed when all *N* phases are completed, i.e. when all inputs have had a chance to choose (successfully or otherwise) an output for transmission during T_k.

[0024] While *N* time slots are required to complete the *N* phases of a given schedule, *N* phases of *N* different schedules may be completed within one time slot using a pipeline approach, by computing the *N* schedules in parallel. But this is effectively equivalent to the completion of one schedule every time slot. In RRGS, a specific future time slot is made available to the inputs for scheduling in a round-robin fashion. An input *i* that starts a schedule for the *k*th time slot (in the future), chooses an output in a round robin (RR) fashion, and sends to the next input $(i + 1) \bmod N$, the set O_k that indicates the remaining output ports that are still free to receive packets during the *k*th time slot. Any input *i* that receives from the previous input $(i - 1) \bmod N$, set O_k of available outputs for the *k*th time slot, chooses one output if possible from this set, and sends to the next input $(i + 1) \bmod N$ the modified set O_k, if input *i* did not complete the schedule for the *k*th time slot, T_k. An input *i* that completes a schedule for the kth time slot should not forward the modified set O_k to the next input $(i + 1) \bmod N$. Thus input $(i + 1) \bmod N$ which did not receive the set O_k in the current time slot, will be starting a new schedule (for a new time slot) in the next time slot. Step 1 of RRGS implies that an input refrains from forwarding the set O_k once in *N* time slots. Input *i* that does not forward the set O_k should be the last one that chooses an output for the *k*th time slot.

[0025] **Theorem 1:** If input $(const - k) \bmod N$ refrains from forwarding the set O_k in the $(k - 1)$th time slot, and the number of inputs *N*, is odd, then input $(const - k) \bmod N$ completes the schedule for the *k*th time slot.

[0026] Proof: The above theorem implies that:

- In every time slot, all *N* inputs will have an opportunity to schedule transmission for a future time slot.
- In each time slot, an input can schedule a transmission for no more than one future time slot.
- In each time slot, an output can be scheduled to receive transmission from only one input.

[0027] Fix input $i = (const - k) \bmod N$ that refrains from forwarding the set O_k in the $(k - 1)$th time slot. In such a case, each of the previous $N - 1$ inputs must forward set O_k when it makes a reservation for the *k*th time slot. Note that in the $(k-1-j)$ the time slot, an input $(i + j) \bmod N$ does not forward set O_l to the next input. Also, input $(i - j) \bmod N$ makes a reservation for the *k*th time slot. Such a schedule is feasible if:

$$\forall (1 \le j \le (N-1)) i - j \ne i + j \bmod N \Leftrightarrow$$
$$\forall (1 \le j \le (N-1)) 2.j \ne 0 \bmod N \Leftrightarrow$$
$$N \text{ is odd number}$$

[0028] Schedule for the *k*th time slot was started by the input $(i + 1) \bmod N$ in the $(k - N)$th time slot, since the input *i* did not forward set O_{k-N} in the $(k - N - 1)$th time slot.

[0029] The timing diagram for an embodiment with a 5X5 switch is shown in Figure 1. This figure shows the relation between inputs and the time slots for which they are choosing their outputs. For example, in time slot T_5, input I is scheduling or choosing an output for transmission during time slot T_{10} while I_3 is scheduling for T_9 and so on. In the next time slot T_6, I_1 is scheduling for T_8 and so on. A Bold vertical line denotes that the previous input completed a schedule, and the next input will start a new schedule. An input does not forward the set *0* to the next input if it is the last one to choose an output for the associated time slot. Since this condition occurs once per N = 5 time slots, an input makes a decision not to forward the set *0*, by means of a modulo N counter.

[0030] Finally, the actions taken by RRGS in the th time slot (for example, the current time slot). O_k denotes a set of available outputs for the k th time slot. Let, $k(i, l) > 0$ denote the time slot for which input *i* reserves an output in the *l* th time slot, and $i_l\ (const - N - 1) \bmod N$ denotes the input that starts a new schedule in the *l*-th time slot. Also $k(i, l) = 0$ implies that the action of input *i* in time slot *l* is suppressed. The scheduler requires proper initialization. This initialization period lasts for *N* time slots. Assume that the initialization process commences in the first time slot T_1. The initialization process is started by selling $k(0, 1) = k(1, 1) = \ldots = k(N - 1,1) = 0$, const = $N + 1$. That is, the actions of all the inputs are suppressed for the first *N* time slots, unless modified subsequently. It is also assumed that packets are queued at the input in logically separate queues called input-output queues, with one queue for each output port to prevent HOL blocking. Further, receive input-output queues and transmit input-output queues are also provided.

- $O_{l+N}=\{0,1,\ldots,N - 1\}$

- $k((i_l, l) = l + N$, and $k(i, l) = k((i - 1) \bmod N, l - 1)$ for $0 \le i \le N - 1$ and $i \ne i_l$.

- Input i, $0 \le i \le N - 1$, chooses one output j in a RR fashion from the set $O_{k(i,l)}$ for which it has a packet to send, provided that $k(i, l) \ne 0$. (Input h $0 \leqq i \leqq N - 1$ and $i \ne i_l$, received $O_{k(i,l)}$ from the input $(i - 1) \bmod N$ in the previous time slot.) Output j is excluded from $O_{k(i,l)}$.

- Input i, $0 \le i \le N - 1$, stores the address of the chosen output in its connection memory at location $k(i, l) \bmod N$. The head of line packet from the corresponding receive input output queue is moved to the separate transmit input-output queue.

- Input i, $0 \le i \le N - 1$, and $i \ne (i_l - 2) \bmod N$, forwards the set $O_{k(i,l)}$ to the next input $(i + 1) \bmod N$. Note that just N bits of information need be forwarded.

- A cross bar connection is established between the input i, $0 \leqq i \leqq N - 1$; and the output whose address is read from location ($l \bmod N$) of input i's connection memory.

- For each input i, $0 \le i \le N - 1$, the HOL packet of the scheduled transmit input-output queue is transmitted through the switch core.

3. Pipelined RRGS for Even Number of Inputs

[0031] In the scheme for odd number of inputs, each input refrains from forwarding the modified set O_k to its next neighbor, if the input was the last one to choose an output for a future time slot, and thus completed the schedule for that time slot. A direct application of such an algorithm (developed for an odd value of N), will result in some inputs scheduling for more than one future time slot, while other inputs do not schedule at all. Thus to control a switch with an even number of inputs, the pipeline technique is modified.

[0032] Proof of the Theorem 1 infers that delaying instead of blocking the control information would imply even number of inputs. For the case with even number of inputs, each input refrains from forwarding the set O_l to the next input once in N time slots, and, in the next time slot the input forwards the delayed set O_l from the previous time slot. When input i forwards the delayed set O_l, it will not forward current set O_k. Therefore, input i should be the last one that chooses an output for the kth time slot.

[0033] **Theorem 2:** If input $(const - k) \bmod N$ delays the set O_l in the $(k - 2)$th time slot (and forwards it in the $(k - 1)$th time slot), and the number of inputs N, is even, then input $(const - k) \bmod N$ completes the schedule for the kth time slot.

[0034] Proof: Fix the input i $(const - k) \bmod N$ that delays set Olin the $(k\text{-}2)th$ time slot, and forwards the delayed set O_l instead of O_k in the $(k - 1)$th time slot. So, in the $(k - 1 - j)th$ time slot, input $(i + J - 1) \bmod N$ delays set O_m, and input $(i + j) \bmod N$ forwards delayed set O_n. In the $(k - 1 - j)th$ time slot, input $(i - j) \bmod N$ makes a reservation for the kth time slot and forwards O_k provided that $i - j \ne i + j \bmod N$ and $i - j \ne i + j - 1 \bmod N$ which is true for $\le j \le N/2 - 1$ and N even. Input $(i - N/2) \bmod N$ stores O_k during the $(k - 1 - N/2)$th time slot so that no input reserves the kth time slot. In the $(k - 2 - N/2)$th time slot, input $(i - N/2)$ makes a reservation for the kth time slot. In the $(k - 1 - j)$th time slot, $N/2 + 2 \le j \le N$, input $(i - j + 1) \bmod N$ makes a reservation for the kth time slot and forwards O_k, provided that $i - j + 1 \ne i + j \bmod N$ and $i - j + 1 \ne i + j - 1 \bmod N$ which is true for N even.

[0035] So, a schedule for the kth time slot progresses through the pipeline without being interrupted before it is completed by user i. This schedule was started by the user $(i + 1) \bmod N$ in the $(k - N - 1)$th time slot, since the input i delayed the control information in the $(k - N - 2)$th time slot.

[0036] The timing diagram for an embodiment with a 4X4 switch is shown in Figure 2 to illustrate the case with even number of inputs. Shaded rectangle denotes a delaying of the control information 0. Bold vertical line denotes that the previous input completed a schedule, and the next input will start a new schedule.

[0037] Again, the actions that the RRGS takes in the lth time slot for even number of inputs are specified. O_k denotes a set of available outputs for the kth time slot. Let, $k(i, l) > 0$ denote the time slot for which input i reserves an output in the l th time slot, and $i_l = (const - N - 1 - l) \bmod N$ denotes the input that starts a new schedule in the lth time slot. Let $k(i, l) = 0$ imply that the action of input i in time slot l is suppressed. The scheduler requires proper initialization. This initialization period lasts for N time slots. Assume that the initialization process commences in the first time slot T_1. The initialization process is started by setting $k(0, 1) = k(1, 1) = \ldots = k(N - 1, 1)$ 0, $const = N + 2$. That is, the actions of all the inputs are suppressed for the first N time slots, unless modified subsequently.

- $O_{l+N+j} = \{0, 1, \ldots, N - 1\}$

- $k(i_l, l) = l + N + 1$, $k(\text{mod}((i_l+1) \bmod N, l) = k((i_l, l - 2)$, and $k(i, l) = k((i - 1) \bmod N, l - 1)$ for $0 \le i \le N - 1$ and $i \notin \{(i_l, (i_l + 1) \bmod N\}$.

- Input i, $0 \leqq i \leqq N - 1$, chooses one output j in a RR fashion from the set $O_{k(i,l)}$ for which it has a packet to send, provided that $k(i, l) \ne 0$. (Input i $0 \le i \le N - 1$ and $i \ne i_l$, received set $O_{k(i,l)}$ from the input $(i - 1) \bmod N$ in the previous time slot.) Output j is excluded from set $O_{k(i,l)}$.

- Input i, $0 \le i \le N - 1$, stores the address of the chosen output in its connection memory at location $k(i, l)$ mod $(N + 1)$. HOL packet from the corresponding receive input-output queue is moved to the separate transmit input-output queue.

- Input i, $0 \le i \le N - 1$ and $i = (i_l - 2)$ mod N, forwards the set $O_{k(i,l)}$ to the next input $(i + 1)$ mod N. Input $((i_l - 2)$ mod N delays the set $O_{k((il-2)\mathrm{mod}N,\, l)}$ for one time slot before forwarding it.

- A cross bar connection is established between the input i, $0 \le i \le N - 1$ and the output 4 whose address is read from location $(l$ mod $(N + 1))$ of the input i's connection memory.

- For each input i, $0 \le i \le N - 1$, the reserved packet at the head of the scheduled transmit input-output queue is transmitted through the switch core.

4. Multicast Scheduling

[0038] Another aspect of this invention is the incorporation of a multicast function. Multicast packets are stored in a separate queue which is served in first come first served (FCFS fashion). Each queue has a multicast bit map (BM) which denotes the destinations of its HOL packet. In the simplest version, the multicast queue will have priority over the unicast queues. Additional multicast actions that are taken in the *lth* time slot are as follows.

- Input i, $0 \le i \le N - 1$, chooses all outputs j such that $j \in O_{k(i,l)} \cap BM_i$. HOL multicast packet will be transmitted to the chosen outputs in the kth time slot.
- If $O_{k(i,l)} \cap BM_i$ is an empty set, the unicast queues are served. Otherwise, the chosen outputs are excluded from $O_{k(i,l)}$ and BM_i.
- If BM_i is empty, the HOL multicast packets is deleted from the multicast queue.

5. Implementation of the Switch Controller

[0039] Controller for an *NXN* optical cross bar switch is shown in Figure 3. Each input requires an input module (3.11,3.21,3.31...3.N1), a RR arbiter and pipeline controller (3.12,3.22,3.32...3.N2), and a connection memory (3.13,3.23,3.33... 3.N3). The input module (IM) stores incoming packets in logically separate receive queues, with each queue destined for a particular output. The IM sends requests to its associated RR arbiter. The RR arbiter chooses one of the free outputs in a future time slot, and informs the corresponding input and the pipeline controller about this choice. Note that the initialization procedure for the pipeline ensures that in any given time slot, no two arbiters pick the same future time slot for scheduling transmission. The input module stores the successful packet in a separate transmit input-output queue. The RR arbiter also writes its scheduling decision into a specified memory location of the associated connection memory. The location in memory is determined by the time slot for which the packet is scheduled. The pipeline controller informs the RR arbiter of the next input, all the outputs that have not been reserved for a particular time slot; more precisely, it inhibits requests for these reserved outputs. If some input does not forward the control information, its pipeline controller allows the RR arbiter of the next input to be able to choose any output for a future time slot.

[0040] Based on the schedule written in the connection memory, packets from the input modules are transferred to output modules via the switch core.

6. PERFORMANCE COMPARISON

[0041] In this section, iRRGS is compared with other protocols of similar complexity. Complexity is measured by the time that the protocol requires to complete one schedule. HOL, I-TDMA, SLIP, RGS and RRGS protocols are compared. *See* K. Bogineni, K. M. Sivilingam, and P. W. Dowd, "Low-complexity multiple access protocols for wavelength-division multiplexed photonic networks," *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 4, May 1993, pp. 590-604; D. Guo, Y. Yemini, Z. Zhang, "Scalable high-speed protocols for WDM optical star networks," *IEEE INFOCOM'94;* M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input vs. output queuing on a space- division packet switch," *IEEE Transactions on Communications*, vol. COM-35, no. 12, December 1987, pp. 1347-1356; and N. McKeown, P. Varaiya, J. Walrand, "Scheduling cells in an input-queued switch," *Electronic Letters*, vol. 29, no. 25, December 1993, pp. 2174-2175.

[0042] The head of line (HOL) protocol denotes the simplest protocol for the switch with input queuing, *see* M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input vs. output queuing on a space- division packet switch," *IEEE Transactions on Communications*, vol. COM-35, no. 12, December 1987, pp. 1347-1356. Each input sends a request for transmission of the HOL packet, to the appropriate output. The requested output issues a grant to one of the inputs in a round-robin fashion. In the next time slot, the granted inputs send packets to the corresponding outputs.

[0043] In interleaved TDMA (I-TDMA) outputs are assigned to the inputs in a fixed manner, *see* K. Bogineni, K. M. Sivilingam, and P. W. Dowd, "Low-complexity multiple access protocols for wavelength-division multiplexed photonic networks," *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 4, May 1993, pp. 590-604. Time is divded into frames. The transmission schedule is predetermined in each time slot of the frame. Packets are stored in separate queues according to their destinations, and transmitted in their scheduled time slots.

[0044] Iterative round-robin matching with slip

(SLIP) has been proposed in, *see* N. McKeown, P. Varaiya, J. Walrand, "Scheduling cells in an input-queued switch," *Electronic Letters*, vol. 29, no. 25, December 1993, pp. 2174-2175. Each input sends requests to all outputs for which it has packets to send. A requested output issues a grant to one of the requesting inputs, in a round-robin fashion. Inputs that receive multiple grants, choose one of the permitted outputs in a round-robin fashion. Each round-robin choice starts from the position after the last chosen candidate.

[0045] Random greedy scheduling (RGS) is similar to RRGS except that the round-robin choice is replaced by random choice. *see* D. Guo, Y. Yemini, Z. Zhang, "Scalable high-speed protocols for WDM optical star networks," *IEEE INFOCOM'94*. The controller randomly chooses a sequence of inputs, and randomly matches them to unmatched outputs. At high speeds, the RGS cannot be completed within one time slot; however, we assess its performance to investigate the effect of replacing the random choice by a round-robin choice.

[0046] In FIG.4 the average packet delay versus the offered load for HOL, I-TDMA,SLIP, RGS and RRGS are plotted. The analytical performance results obtained agrees well with simulation results.

[0047] FIG.5 shows the complementary distribution function of packet delay in I-TDMA,SLIP, RGS and RRGS for fixed offered traffic. The plotted curves are obtained using simulation results. RRGS significantly outperforms I-TDMA and SLIP for most traffic loads.

[0048] FIG.6 shows protocol performance for non-uniform traffic for a specific traffic matrix for four different loads G_1, G_2, G_3 and G_4.

7. CONCLUSION

[0049] The present invention proposes a pipelined round robin scheduler for fast input buffered packet switches. The RRGS protocol of the present invention provides shorter average packet delay than other protocols of comparable complexity. Packet delay distribution does not exhibit heavy tail. Under non-uniform traffic loading, lightly loaded queues experience longer delay, but, heavily loaded queues experience delays that are significantly lower, compared to delays in other protocols.

[0050] Other modifications and variations to the invention will be apparent to those skilled in the art from the foregoing disclosure and teachings. Thus, while only certain embodiments of the invention have been specifically described herein, it will be apparent that numerous modifications may be made thereto without departing from the spirit and scope of the invention.

Claims

1. A method for determining a time slot in a NxN crossbar switch for a round robin greedy scheduling protocol, comprising N logical queues at each input, corresponding to N output ports, the input for the protocol being a state of all the input-output queues, output of the protocol being a schedule, the method comprising:

   a) choosing input corresponding to i = (constant-k-1) mod N;

   b) stopping if there are no more inputs, otherwise choosing the next input in a round robin fashion determined by i = (i + 1) mod N;

   c) choosing an output j such that a pair (i,j) to a set C= {(i,j) | there is at least one packet from I to j} , if the pair (i,j) exists;

   d) removing i from a set of inputs and going to step b if the pair (i,j) does not exist in step c;

   e) removing i from the set of inputs and j from a set of outputs; and

   f) adding the pair (i,j) to the schedule and going to step b.

2. A method of scheduling wherein in each time slot N distinct schedules are in progress simultaneously for N distant future time slots, the method comprising;

   a) making a specific future time slot available to input for scheduling in a round-robin fashion;
   b) selecting an output for a kth time slot in future, by an input i;
   c) starting a schedule for the kth time slot in future;
   d) determining a next input (i+1) mod N and sending to the next input remaining outputs that are free to receive packets during the kth time slot.

3. The method of claim 2 wherein if an input I did not complete a schedule for the kth time slot selects an output and sends a modified output set to a next input ; and if the input I completes a schedule for the kth time slot it does not send the output set to the next input.

4. The method of claim 3 wherein an input that does not receive a modified set of outputs from a previous input starts a new schedule.

5. A method of pipelined round robin greedy scheduling for odd number of inputs wherein an lth time slot is completed using a process comprising:

   e) Initializing k(0,1)=k(1,1)=...k(N-1,1)=0, const = N+1, wherein, k(i, *l*) > 0 is the time slot for which input i reserves an output in ıth time slot, i_l=(const - N - 1) mod N denotes an input that starts a new schedule in ıth time slot, and k(i, *l*) = 0 implies that the action of input i in time slot *l* is suppressed;

f) selling O_{l+N}={0,1, . . .,N—1}, k(i_l, *l*) = l+N, *k(i, l) = k((i* — 1) mod *N, l* — 1) for 0 ≤ *i* ≤ *N*—1 and *i* ≠ i_l;

g) choosing one output *j* at input *i*, 0 ≦ *i* ≦ *N* - 1, in a round-robin fashion from the set $O_{k(i,\,l)}$ for which it has a packet to send, provided that *k(i, 1)* ≠ 0 and excluding j from $O_{k(i,\,l)}$.;

h) storing the address, at input *i*, 0 ≤ *i* ≤ *N* — 1, of the chosen output in its connection memory at location *k(i, l)* mod *N* and moving a head of line (HOL) packet from a corresponding receive input-output queue to separate transmit input-output queue;

i) forwarding the set $O_{k(i,l)}$ at Input *i*, 0 ≤ *i* ≤ *N* — 1 and i ≠ (i_l—2) mod *N*,to the next input (*i* + 1) mod *N;*

j) establishing a cross bar connection between the input *i* 0 ≤ *i* ≤ *N* — 1 and output whose address is read from location (*l* mod (*N* + 1)) of the input *i*'s connection memory; and

k) transmitting the reserved packet at the head of the scheduled transmit input-output queue i through the switch core for each input *i*, 0 ≤ *i* ≤ *N* — 1.

6. A method of pipelined round robin greedy scheduling for even number of inputs wherein an lth time slot is completed using a process comprising:

l) Initializing k(0,1)=k(1,1)=...k(N-1,1)=0, const = N+1, wherein, k(i,*l*) > 0 is the time slot for which input i reserves an output in *l* th time slot, i_l=(const - N - 1) mod N denotes an input that starts a new schedule in *l* th time slot, and k(i, *l*) = 0 implies that the action of input i in time slot *l* is suppressed;

m) setting O_{l+N}={0,1, . . .,N—1}, k(i_l, *l*) = *l*+N+1, k(mod(i_l+1) mod N, *l*) = k(i_l, *l*—2), and k(i, *l*) = k((i—1) mod *N, l*— 1) for 0 ≤ *i* ≤ N — 1 and(i_l+ 1) mod *N)* and *i* ∉ {(i_l, (i_l + 1) mod *N};*

n) choosing one output *j* at input *i*, 0 ≦ *i* ≦ N — 1, in a round-robin fashion from the set $O_{k(i,\,l)}$ for which it has a packet to send, provided that *k(i,* 1) ≠ 0 and excluding j from $O_{k(i,,l)}$;

o) storing the address, at input *i*, 0 ≤ *i* ≤ *N* — 1, of the chosen output in its connection memory at location *k(i, l)* mod (*N* + 1) and moving an HOL packet from a corresponding receive input-output queue to separate transmit input-output queue;

p) forwarding the set $O_{k(i,l)}$ at Input *i*, 0 ≤ *i* ≤ *N* — 1 and *i*= (i_l — 2) mod *N*,to the next input (*i* + 1) mod *N*, wherein Input (i_l — 2) mod *N* delays the set $O_{k((i_l-2)\mathrm{modN},.)}$ for one time slot before forwarding it;

q) Establishing a cross bar connection between the input *i*, 0 ≤ *i* ≤ *N* — 1 and output whose address is read from location (*l* mod (*N* + 1)) of the input *i*'s connection memory;

r) transmitting the reserved packet at the head of the scheduled transmit input-output queue is transmitted through the switch core for each input *i*, 0 ≤ *i* ≤ *N* — 1.

7. A method according to claim 5 wherein multicast scheduling is incorporated in round-robin greedy scheduling algorithm wherein multicast packets are stored in a first come first served fashion and have priority over unicast queues and steps in lth time slot further comprise:

s) choosing all outputs *j* such that *j* ∈ $O_{k(i,\,,l)}$ ∩ BM_i at input *i*, 0 ≤ *i* ≤ *N* — 1 and transmitting HOL multicast packets the chosen outputs in the *k*th time slot.

t) serving unicast queues if $O_{k(i,\,,l)}$ ∩ BM_i is an empty set, otherwise excluding the chosen outputs from $O_{k(i,\,,l)}$ and BM_i.

u) deleting HOL multicast packets from the multicast queue If BM_i is empty.

8. A method according to claim 6 wherein multicast scheduling is incorporated in round-robin greedy scheduling algorithm wherein multicast packets are stored in a first come first served fashion and have priority over unicast queues and steps in lth time slot further comprise:

v) chooses all outputs *j* such that *j* ∈ $O_{k(i,l)}$ ∩ BM_i at input *i*, 0 ≤ *i* ≤ *N* — 1 and

w) transmitting HOL multicast packets the chosen outputs in the *k*th time slot.

x) serving unicast queues if $O_{k(i,l)}$ ∩ BM_i is an empty set, otherwise excluding the chosen outputs from $O_{k(i,,l)}$ and BM_i.

y) deleting HOL multicast packets from the multicast queue If BM_i is empty.

9. An N stage pipeline system for scheduling a NxN switch where a stage i is associated with an input I, said stage I schedules transmission to an output in a future time slot, said future time slot rippling through all stages, wherein all pipeline stages corresponding to inputs are performing scheduling concurrently such that no two inputs choose a same future time slot at a same time, output slots being selected based on a round-robin fashion, wherein when an output is selected by a stage, the output is removed from a free pool of outputs such that a pipeline stage does not pick the output that has already been selected during a time slot by inputs.

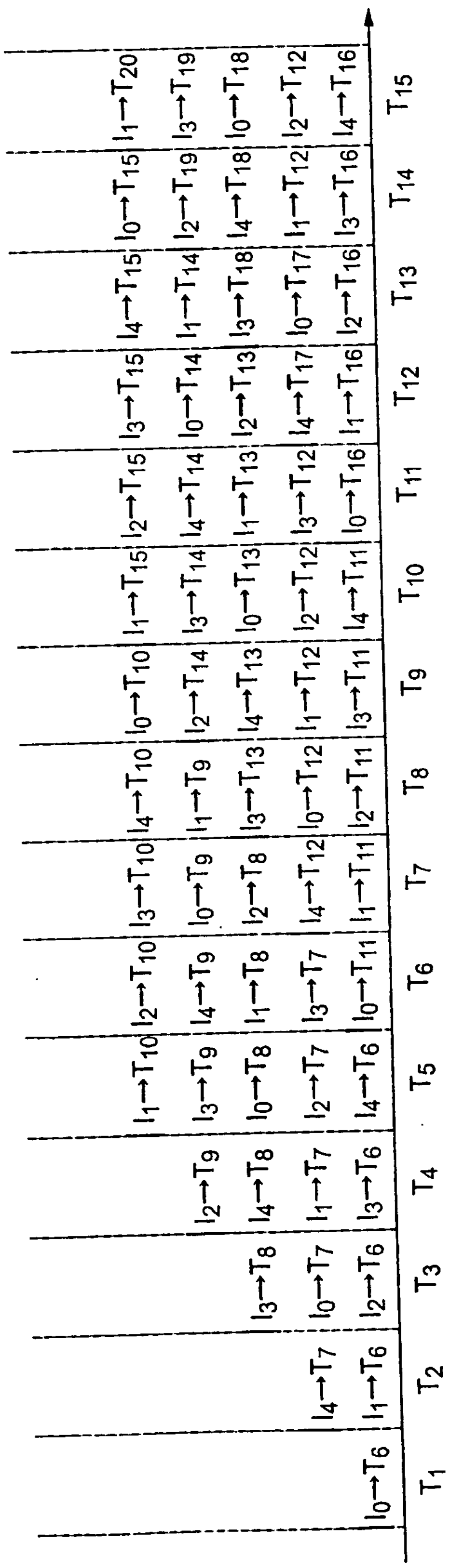

$I_0 \rightarrow T_6$
$I_4 \rightarrow T_7$
$I_1 \rightarrow T_6$
$I_3 \rightarrow T_8$
$I_0 \rightarrow T_7$
$I_2 \rightarrow T_6$
$I_2 \rightarrow T_9$
$I_4 \rightarrow T_8$
$I_1 \rightarrow T_7$
$I_3 \rightarrow T_6$
$I_1 \rightarrow T_{10}$
$I_3 \rightarrow T_9$
$I_0 \rightarrow T_8$
$I_2 \rightarrow T_7$
$I_4 \rightarrow T_6$
$I_2 \rightarrow T_{10}$
$I_4 \rightarrow T_9$
$I_1 \rightarrow T_8$
$I_3 \rightarrow T_7$
$I_0 \rightarrow T_{11}$
$I_3 \rightarrow T_{10}$
$I_0 \rightarrow T_9$
$I_2 \rightarrow T_8$
$I_4 \rightarrow T_{12}$
$I_1 \rightarrow T_{11}$
$I_4 \rightarrow T_{10}$
$I_1 \rightarrow T_9$
$I_3 \rightarrow T_{13}$
$I_0 \rightarrow T_{12}$
$I_2 \rightarrow T_{11}$
$I_0 \rightarrow T_{10}$
$I_2 \rightarrow T_{14}$
$I_4 \rightarrow T_{13}$
$I_1 \rightarrow T_{12}$
$I_3 \rightarrow T_{11}$
$I_1 \rightarrow T_{15}$
$I_3 \rightarrow T_{14}$
$I_0 \rightarrow T_{13}$
$I_2 \rightarrow T_{12}$
$I_4 \rightarrow T_{11}$
$I_2 \rightarrow T_{15}$
$I_4 \rightarrow T_{14}$
$I_1 \rightarrow T_{13}$
$I_3 \rightarrow T_{12}$
$I_0 \rightarrow T_{16}$
$I_3 \rightarrow T_{15}$
$I_0 \rightarrow T_{14}$
$I_2 \rightarrow T_{13}$
$I_4 \rightarrow T_{17}$
$I_1 \rightarrow T_{16}$
$I_4 \rightarrow T_{15}$
$I_1 \rightarrow T_{14}$
$I_3 \rightarrow T_{18}$
$I_0 \rightarrow T_{17}$
$I_2 \rightarrow T_{16}$
$I_0 \rightarrow T_{15}$
$I_2 \rightarrow T_{19}$
$I_4 \rightarrow T_{18}$
$I_1 \rightarrow T_{12}$
$I_3 \rightarrow T_{16}$
$I_1 \rightarrow T_{20}$
$I_3 \rightarrow T_{19}$
$I_0 \rightarrow T_{18}$
$I_2 \rightarrow T_{12}$
$I_4 \rightarrow T_{16}$
T_1
T_2
T_3
T_4
T_5
T_6
T_7
T_8
T_9
T_{10}
T_{11}
T_{12}
T_{13}
T_{14}
T_{15}

FIG.1

FIG.2

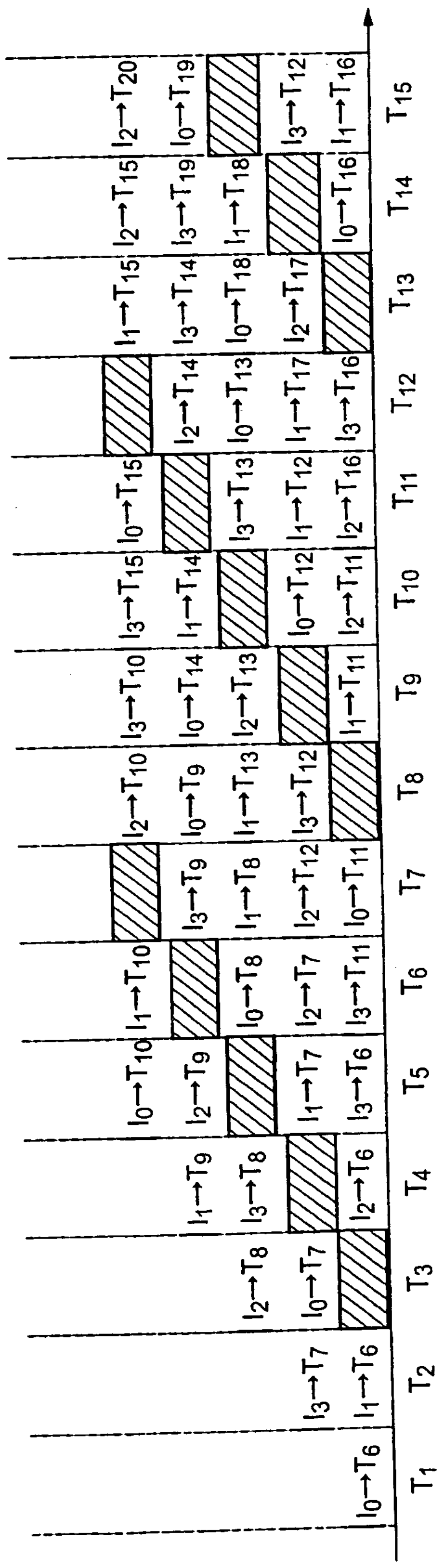

$I_0 \rightarrow T_6$
$I_3 \rightarrow T_7$
$I_1 \rightarrow T_6$
$I_2 \rightarrow T_8$
$I_0 \rightarrow T_7$
$I_1 \rightarrow T_9$
$I_3 \rightarrow T_8$
$I_2 \rightarrow T_6$
$I_0 \rightarrow T_{10}$
$I_2 \rightarrow T_9$
$I_1 \rightarrow T_7$
$I_3 \rightarrow T_6$
$I_1 \rightarrow T_{10}$
$I_0 \rightarrow T_8$
$I_2 \rightarrow T_7$
$I_3 \rightarrow T_{11}$
$I_3 \rightarrow T_9$
$I_1 \rightarrow T_8$
$I_2 \rightarrow T_{12}$
$I_0 \rightarrow T_{11}$
$I_2 \rightarrow T_{10}$
$I_0 \rightarrow T_9$
$I_1 \rightarrow T_{13}$
$I_3 \rightarrow T_{12}$
$I_3 \rightarrow T_{10}$
$I_0 \rightarrow T_{14}$
$I_2 \rightarrow T_{13}$
$I_1 \rightarrow T_{11}$
$I_3 \rightarrow T_{15}$
$I_1 \rightarrow T_{14}$
$I_0 \rightarrow T_{12}$
$I_2 \rightarrow T_{11}$
$I_0 \rightarrow T_{15}$
$I_3 \rightarrow T_{13}$
$I_1 \rightarrow T_{12}$
$I_2 \rightarrow T_{16}$
$I_2 \rightarrow T_{14}$
$I_0 \rightarrow T_{13}$
$I_1 \rightarrow T_{17}$
$I_3 \rightarrow T_{16}$
$I_1 \rightarrow T_{15}$
$I_3 \rightarrow T_{14}$
$I_0 \rightarrow T_{18}$
$I_2 \rightarrow T_{17}$
$I_2 \rightarrow T_{15}$
$I_3 \rightarrow T_{19}$
$I_1 \rightarrow T_{18}$
$I_0 \rightarrow T_{16}$
$I_2 \rightarrow T_{20}$
$I_0 \rightarrow T_{19}$
$I_3 \rightarrow T_{12}$
$I_1 \rightarrow T_{16}$
T_1
T_2
T_3
T_4
T_5
T_6
T_7
T_8
T_9
T_{10}
T_{11}
T_{12}
T_{13}
T_{14}
T_{15}

FIG.3

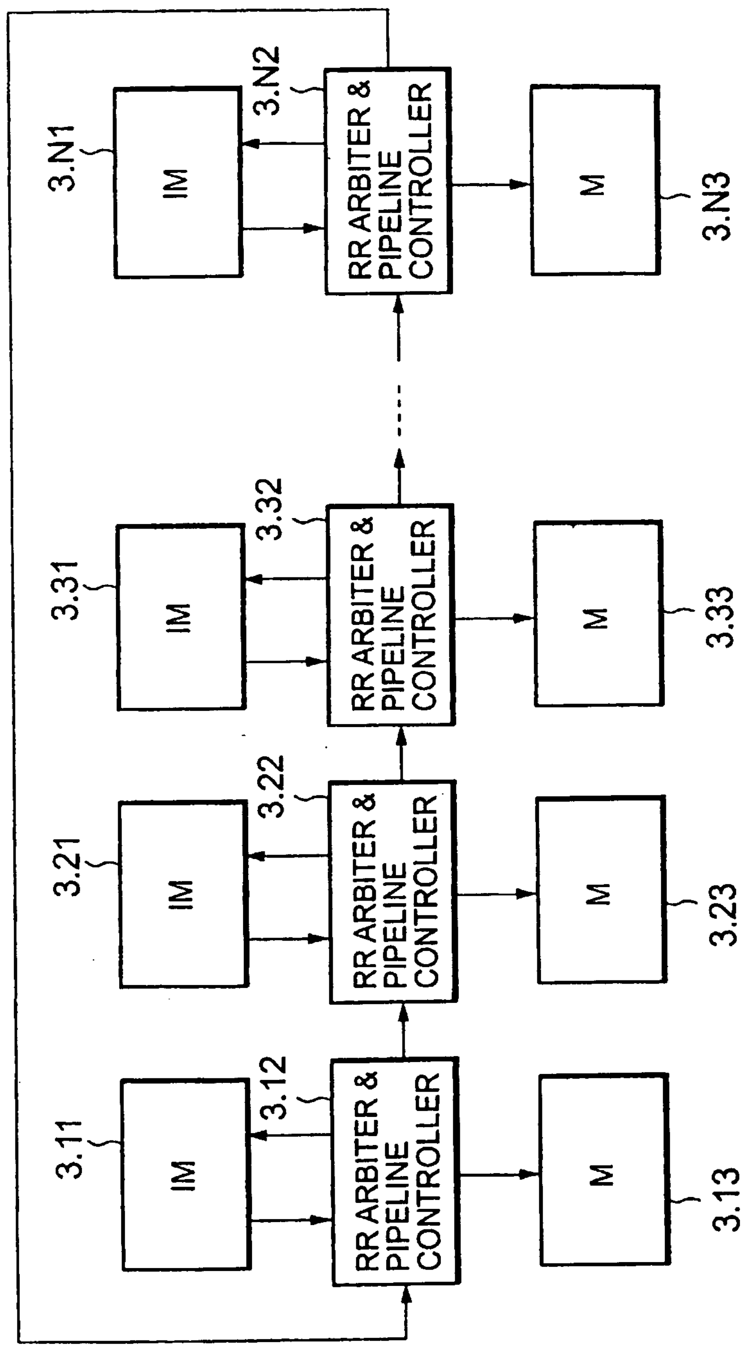

IM
IM
IM
IM
RR ARBITER & PIPELINE CONTROLLER
RR ARBITER & PIPELINE CONTROLLER
RR ARBITER & PIPELINE CONTROLLER
RR ARBITER & PIPELINE CONTROLLER
M
M
M
M
3.11
3.12
3.13
3.21
3.22
3.23
3.31
3.32
3.33
3.N1
3.N2
3.N3

FIG.4

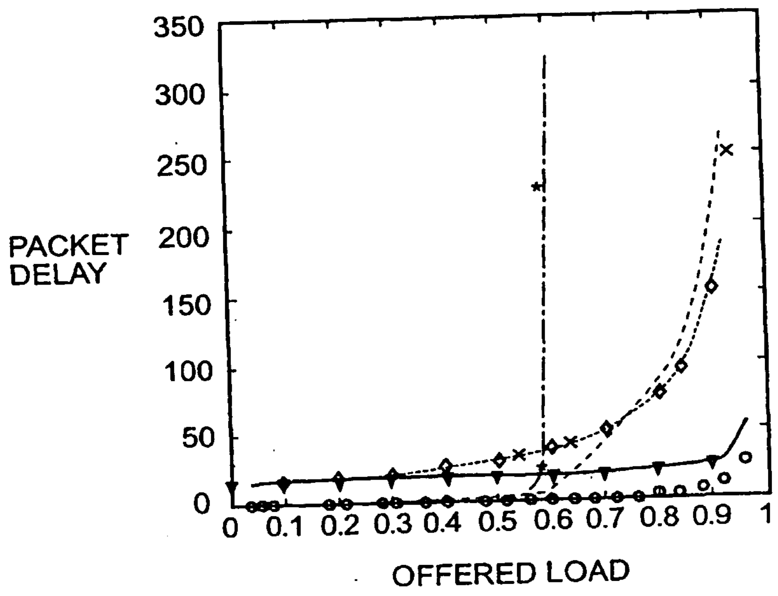

PACKET DELAY
350
300
250
200
150
100
50
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
OFFERED LOAD

FIG.5A

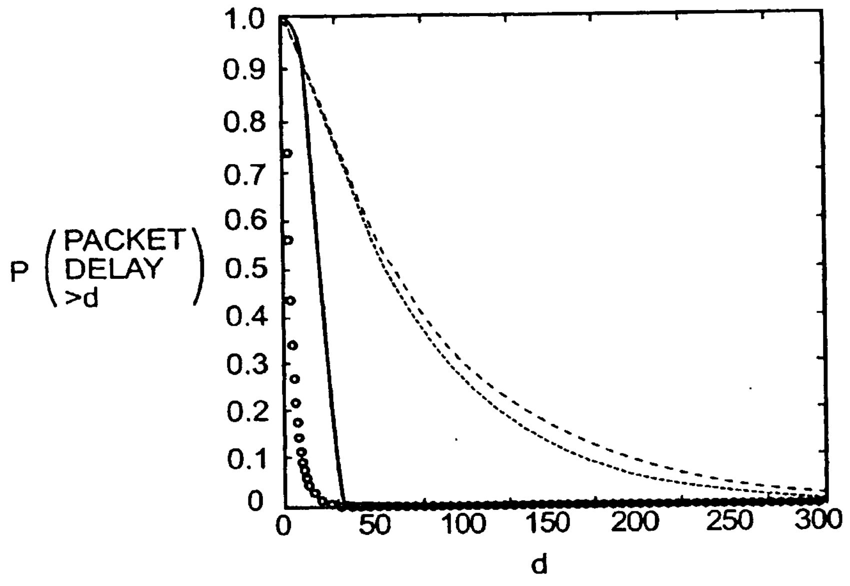

1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
P (PACKET DELAY >d)
0
50
100
150
200
250
300
d

FIG.5B

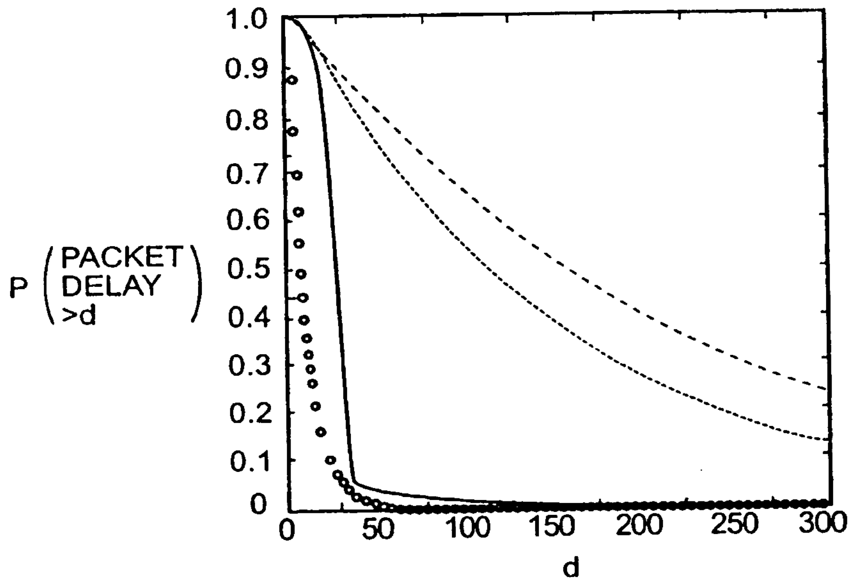

1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
P (PACKET DELAY >d)
0
50
100
150
200
250
300
d

FIG.6A

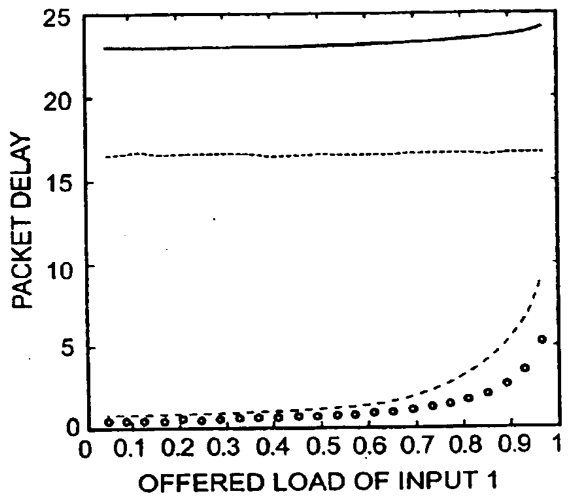


FIG.6B

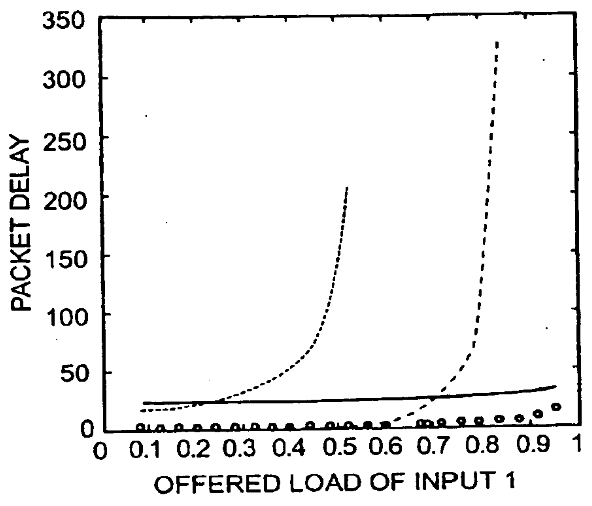


FIG.6C

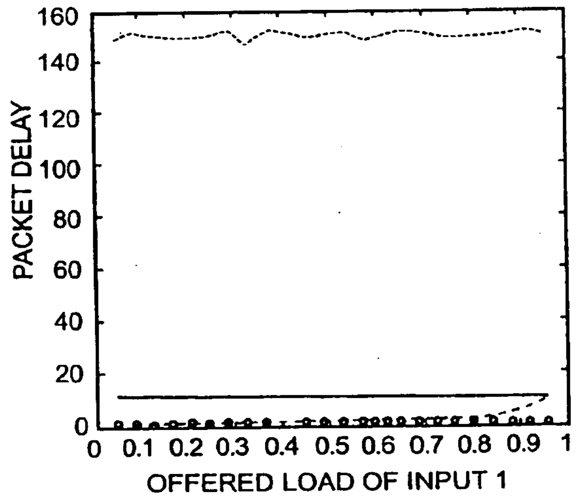
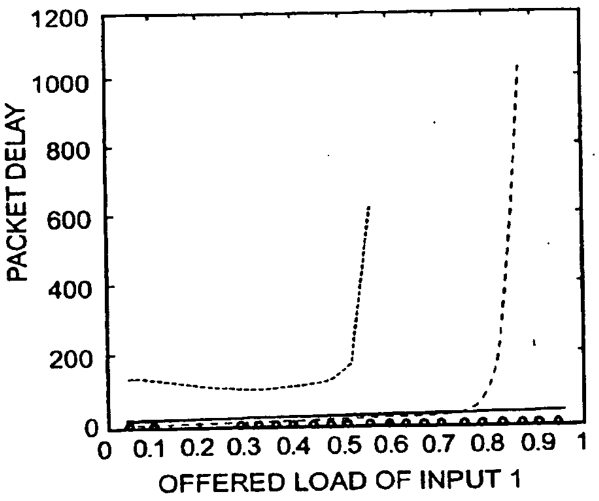


FIG.6D


PACKET DELAY
OFFERED LOAD OF INPUT 1